Tópicos Especiais em Fundamentos da Computação
Shell Scripting

sed

Andrei Rimsa Álvares
andrei@cefetmg.br

Sumário

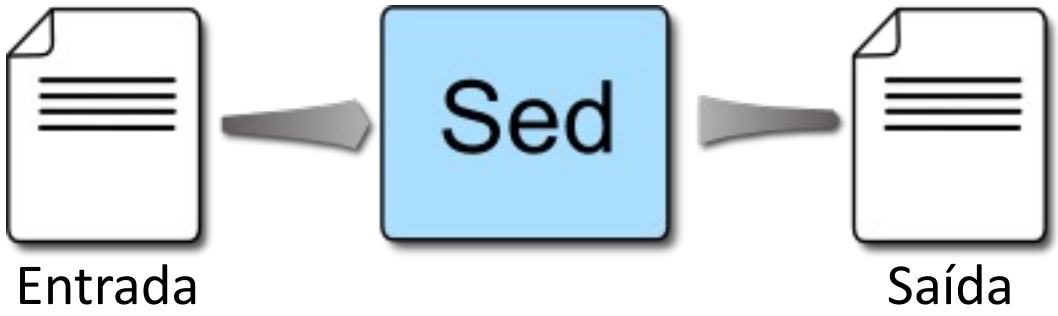
- Introdução
- Visão geral
- Exemplos
- Mais exemplos

INTRODUÇÃO

Introdução

- A ferramenta **sed** (*stream editor*) é um editor de texto em modo *batch* (não interativo)

- Ele transforma um fluxo de dados de entrada que vem de um arquivo ou da entrada padrão
  - Comumente usado em um *pipe* como um filtro



Introdução

- **sed** (não interativo) é mais eficiente que o editor ed (interativo), já que faz apenas uma passada na entrada

sed

ed

Introdução

- sed possui duas famosas implementações
  - GNU sed (usado pela maioria das distribuições Linux)
  - BSD sed (usado por BSDs e Mac OS X)

Os comandos que serão vistos aplicam para ambas as versões!

VISÃO GERAL

Sintaxe

- A linha de comando usando sed tem a seguinte sintaxe

```
sed [-n] program [file-list]
sed [-n] –f program-file [file-list]
```

  - Um programa em **sed** pode ser especificado diretamente na linha de comando (**program**) ou por um arquivo-fonte (**program-file**)
  - **sed** pode processar arquivos especificados em uma lista de nomes via argumentos (**file-list**); caso não seja especificado, a entrada é lida da entrada padrão

Opções

Dica: GNU `sed` possui sintaxe alternativa para essas opções usando dois hífens (--)

- Opções usadas em linha de comando por sed

  `-f program-file`: faz com que leia seu programa do arquivo chamado **`program-file`** ao invés da linha de comando (pode-se usar mais de uma vez)

  `-i`[*`sufixo`*]: Edita arquivos *in-place*. Sem essa opção, sed envia a saída para a saída padrão, com essa opção ele sobrescreve o arquivo que está sendo processado pela saída; quando especificado um sufixo, sed faz um backup do arquivo original adicionando o sufixo ao nome do novo arquivo

  `-n`: faz com que o **`sed`** não copie as linhas do arquivo para a saída original, a não ser aquelas especificadas pela instrução Print (**`p`**)

Noções Básicas

- Um programa em sed consiste de uma ou mais linhas com a seguinte sintaxe

```
[address[,address]] instruction [argument-list]
```

  - O endereço (*address*) é opcional, se for omitido o comando processa todas as linhas da entrada
  - A instrução (*instruction*) é uma instrução de edição que modifica o texto
  - O número e tipos dos argumentos na lista (*argument-list*) depende da instrução

Noções Básicas

- A ferramenta sed processa a entrada da seguinte forma

1) Lê uma linha da entrada (entrada padrão ou *file-list*)
2) Lê a primeira instrução do programa (*program* ou *file-program*) - se o(s) endereço(s) selecionar(em) a linha de entrada age sobre a entrada conforme especificado na entrada
3) Lê a próxima instrução do programa - se o(s) endereço(s) selecionar(em) a linha de entrada age sobre a entrada conforme especificado na entrada (possivelmente na linha manipulada)
4) Repete o passo 3 até executar todas as instruções do programa
5) Recomeça no passo 1 se houver outra linha de entrada, caso contrário o comando termina

Endereços

- Endereços podem ser
  - **Números:** um número de linha é um endereço que seleciona uma linha da entrada; como caso especial $ seleciona a última linha
  - **Expressão regular:** uma expressão regular é um endereço que seleciona uma linha cujo conteúdo casa com a expressão regular
- Zero, um ou dois endereços podem preceder uma instrução
  - **Zero:** se não especificar um endereço, sed selecionará todas
  - **Um:** um endereço faz com que sed processe as linhas selecionadas
  - **Dois:** especificar dois endereços faz com que as instruções processem as linhas em grupos
    - Primeiro: seleciona a primeira linha no primeiro grupo
    - Segundo: seleciona a linha subsequente que case (essa linha é a última do primeiro grupo)

Instruções

- O utilitário sed possui dois buffers, os comandos abaixo funcionam com o **Pattern Space** que inicialmente guarda a linha que acabou de ser lida

  - `a` *(append)*
  - `c` *(change)*
  - `d` *(delete)*
  - `i` *(insert)*
  - `N` *(next without write)*
  - `n` *(next)*
  - `p` *(print)*
  - `q` *(quit)*
  - `r` *file (read)*
  - `s` *(substitute)*
  - `w` *file (write)*

O outro buffer, **Hold Space**, será discutido posteriormente

Instruções

- O utilitário sed possui dois buffers, os comandos abaixo funcionam com o **Pattern Space** que inicialmente guarda a linha que acabou de ser lida
  - **a** (*append*): a instrução de Anexar adiciona uma ou mais linhas na linha selecionada (se usar dois endereços, será adicionada a cada linha selecionada por eles); a instrução **sempre** escreve o texto anexado, mesmo se usar a *flag* **–n** ou remover a linha com **d**

```
[address[,address]] a\
text \
text \
...
text
```

  - **c** (*change*): similar a instrução de Anexar e Inserir, mas altera o texto da linha selecionada (com dois endereços, sed altera a faixa de linhas por uma ocorrência do novo texto)

Instruções

- ...continuação
  - **d** (*delete*): a instrução de Remover faz com que sed não escreva as linhas selecionadas e não termine de processá-las; uma nova linha é lida após essa instrução e o programa começa do início
  - **i** (*insert*): a instrução de Inserir é semelhante a de Anexar e Modificar, a não ser que adiciona o novo texto ANTES da linha selecionada
  - **N** (*next without write*): essa instrução lê a próxima linha da entrada e anexa na linha atual, separadas por uma nova linha (\n)
  - **n** (*next*): essa instrução escreve a linha atualmente selecionada (se apropriado), lê a próxima linha e inicia o processamento dessa com a próxima instrução do programa

Instruções

- ...continuação
  - **`p`** (*print*): essa instrução escreve a linha selecionada na saída padrão imediatamente e não reflete os efeitos de instruções subsequentes (essa instrução sobrescreve a opção –n da linha de comando)
  - **`q`** (*quit*): essa instrução aborta a execução imediatamente
  - **`r file`** (*read*): essa instrução lê o conteúdo do arquivo especificado e o anexa na linha selecionada (um ÚNICO espaço em branco deve ser inserido entre a instrução e o nome do arquivo)
  - **`w file`** (*write*): similar a instrução de Imprimir, mas envia a saída para o arquivo especificado (um ÚNICO espaço em branco deve ser inserido entre a instrução e o nome do arquivo)

Instruções

- ...continuação
  - **s** (*substitute*): essa instrução possui a seguinte sintaxe

    `s/pattern/replacement-string/[g][p][w file]`

    pattern: é uma expressão regular que tradicionalmente é delimitada por barra (/), mas pode usar qualquer caractere que não espaço e nova linha

    replacement-string: começa imediatamente após o segundo delimitador e termina com o mesmo delimitador (o terceiro delimitador é obrigatório); pode conter um & que é substituído pelo padrão casado

    g (*global*): causa a substituição de todas as ocorrências do padrão não sobrepostos

    p (*print*): manda todas as linhas que fazem substituição para a saída

    w (*write*): similar a **p**, mas manda a saída para o arquivo

Estruturas de Controle

- sed suporta as seguintes estruturas de controle
  - **!** (*NOT*)**:** faz com que sed aplique a instrução subsequente, na mesma linha, em cada linha não selecionada pela parte de endereços da instrução
    - **Ex.: `3!d`** remove todas as linhas a não ser a terceira, enquanto **`$!p`** mostra todas as linhas menos a última
  - **{}** (*group instructions*): um grupo de instruções pode ser envolto em chaves, de forma que um endereço ou um par de endereços selecione as linhas nas quais as instruções irão operar; use ponto-vírgula (;) para separar múltiplos comandos em uma única linha

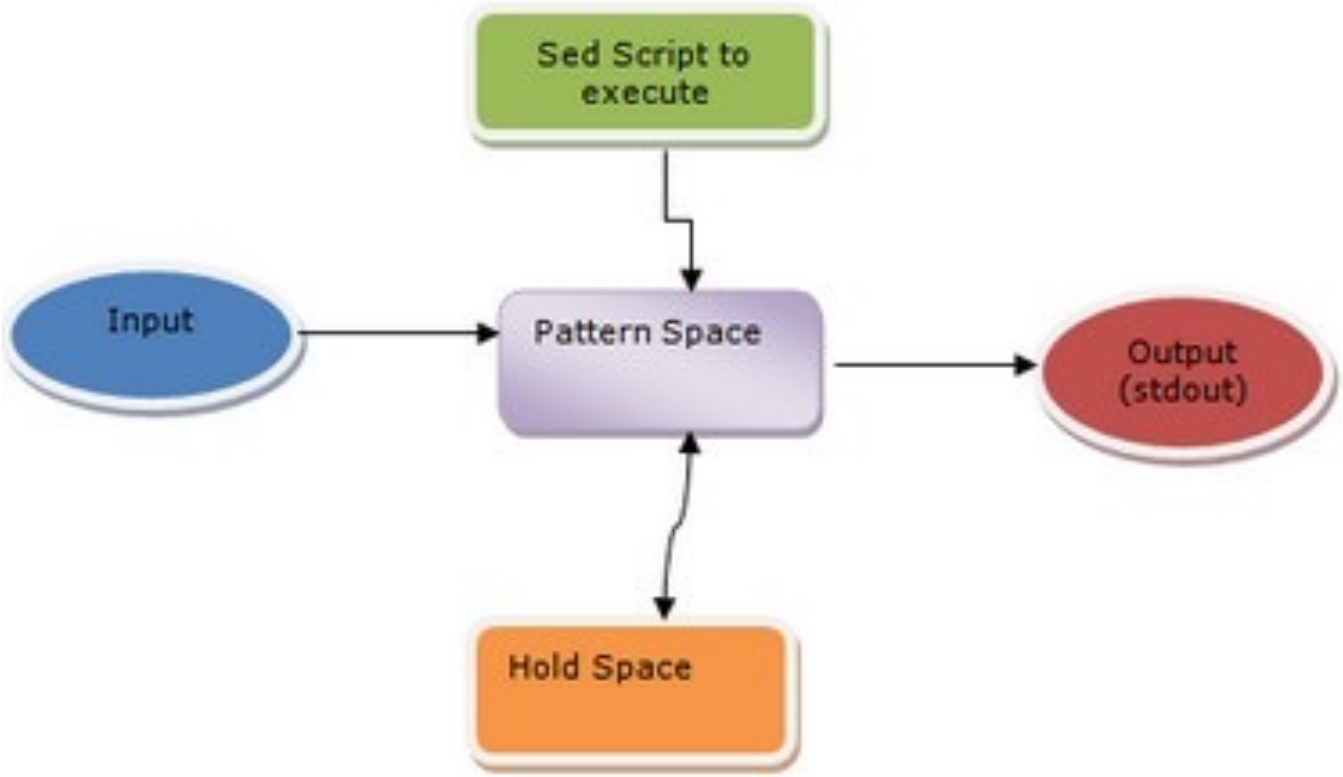
GNU sed suporta instruções de desvio (*branch*), que não serão cobertas

Hold Space

- O **Hold Space** pode armazenar dados enquanto dados são manipulados no **Pattern Space**
  - É um buffer temporário que, até que se insiram dados, começa vazio
- Os comandos a seguir são usados para mover dados entre **Hold Space** e o **Pattern Space**
  - **g:** copia o conteúdo do **Hold space** para o **Pattern Space**; o conteúdo original do **Pattern Space** é perdido
  - **G:** adiciona uma nova linha (\n) e o conteúdo do **Hold Space** para o **Pattern Space**
  - **h:** copia o conteúdo do **Pattern Space** para o **Hold Space**; o conteúdo original do **Hold Space** é perdido
  - **H:** adiciona uma nova linha (\n) e o conteúdo do **Pattern Space** para o **Hold Space**
  - **x:** troca o conteúdo do **Pattern Space** e do **Hold Space**

Resumo

- O diagrama a seguir mostra o funcionamento do sed



EXEMPLOS


space
Scrn
Scroll Lock
Pause
rs112054560
examples
Insert
Home
MediaFocus

Arquivo de Entrada

- Considere o seguinte arquivo que será usado nos próximos exemplos

```
$ cat > lines << EOF
> Line one.
> The second line.
> The third.
> This is line four.
> Five.
> This is the sixth sentence.
> This is line seven.
> Eighth and last.
> EOF
$
```

Instrução de Imprimir (**p**)

- **sed** envia por padrão todas as linhas (selecionadas ou não) para a saída padrão, a não ser que se use a opção **–n** na linha de comando

```
$ sed '/line/ p' lines
Line one.
The second line.
The second line.
The third.
This is line four.
This is line four.
Five.
This is the sixth sentence.
This is line seven.
This is line seven.
Eighth and last.
```

```
$ sed -n '/line/ p' lines
The second line.
This is line four.
This is line seven.
```

Instrução de Imprimir (**p**)

- **sed** pode mostrar parte da entrada baseado nos números das linhas

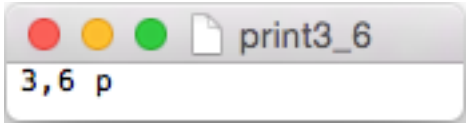
```
$ sed -n '3,6 p' lines
The third.
This is line four.
Five.
This is the sixth sentence.
```

- Pode-se usar a instrução **q** (quit) sem a opção –n na linha de comando para imprimir até determinada linha

```
$ sed '5 q' lines
Line one.
The second line.
The third.
This is line four.
Five.
```

Programa em Arquivo

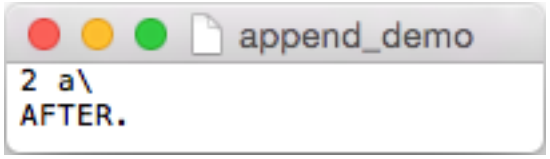
- Quando é preciso dar comandos mais complexo ou com mais instruções, pode-se usar um arquivo de programa (**program-file**); a opção –f instrui o sed a ler o programa do arquivo




```
$ sed -n -f print3_6 lines
The third.
This is line four.
Five.
This is the sixth sentence.
```

Instrução de Anexar (a)

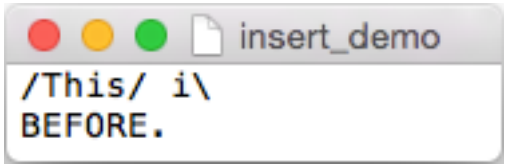
- O programa a seguir seleciona a linha 2 e usa a instrução de Anexar uma nova linha (\n) e o texto "AFTER." à linha selecionada; como não foi usada a opção –n, sed mostra todas as linhas da entrada




```
$ sed -f append_demo lines
Line one.
The second line.
AFTER.
The third.
This is line four.
Five.
This is the sixth sentence.
This is line seven.
Eighth and last.
```

Instrução de Inserir (`i`)

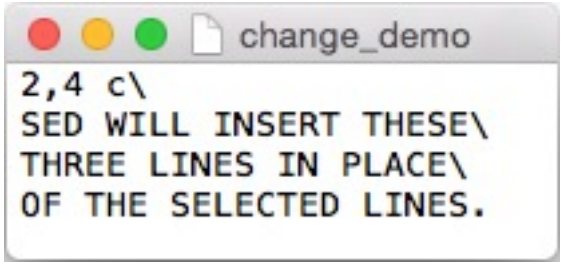
- O programa a seguir seleciona todas as linhas que possuem a string "This" e insere uma nova linha e o texto "BEFORE." antes delas




```
$ sed -f insert_demo lines
Line one.
The second line.
The third.
BEFORE.
This is line four.
Five.
BEFORE.
This is the sixth sentence.
BEFORE.
This is line seven.
Eighth and last.
```

Instrução de Modificar (c)

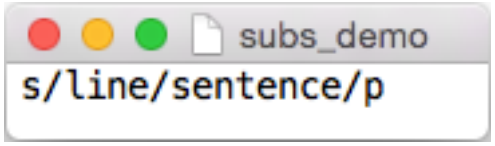
- Esse exemplo mostra a instrução Modificar em uma faixa de endereços (grupo de linhas), onde não é substituída cada linha pelo novo texto, mas sim o bloco de linhas com uma única ocorrência




```
$ sed -f change_demo lines
Line one.
SED WILL INSERT THESE
THREE LINES IN PLACE
OF THE SELECTED LINES.
Five.
This is the sixth sentence.
This is line seven.
Eighth and last.
```

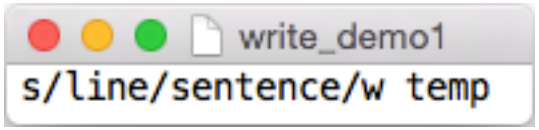
Instrução de Substituição (s)

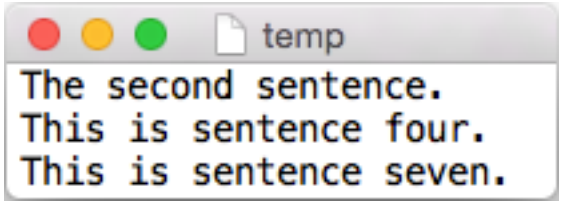
- Nesse exemplo, a instrução de Substituição seleciona todas as linhas já que não possui endereço; em cada linha, o programa substitui a primeira ocorrência de "line" por "sentence"




```
$ sed -n -f subs_demo lines
The second sentence.
This is sentence four.
This is sentence seven.
```

- Exemplo similar ao anterior, mas escreve no arquivo **temp**






```
$ sed -f write_demo1 lines
Line one.
The second sentence.
The third.
This is sentence four.
Five.
This is the sixth sentence.
This is sentence seven.
Eighth and last.
```

Instrução de Substituição (s)

- Exemplo de shell script com múltiplas instruções de substituições simultâneas

sub

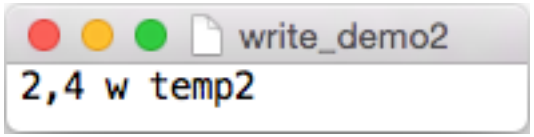
```
#!/bin/bash

for file; do
  echo $file
  mv $file $$.subhld
  sed 's/REPORT/report/g
       s/FILE/file/g
       s/PROCESS/process/g' $$.subhld > $file
done
rm $$.subhld
```

```
$ ./sub file1 file2 file3
file1
file2
file3
```

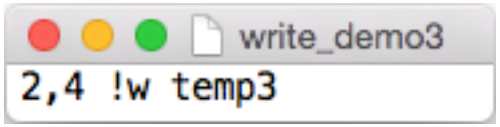
Instrução de Escrita (w)

- Exemplo de instrução de Escrita que copia parte de um arquivo (*files*) para um outro arquivo (*temp2*)




```
$ sed -n -f write_demo2 lines
$ cat temp2
The second line.
The third.
This is line four.
```

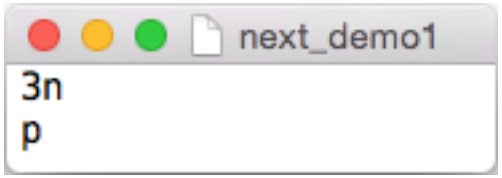
- Exemplo similar ao anterior, mas negando a instrução com **!** (**NOT**)




```
$ sed -n -f write_demo3 lines
$ cat temp3
Line one.
Five.
This is the sixth sentence.
This is line seven.
Eighth and last.
```

Próxima Instrução (n)

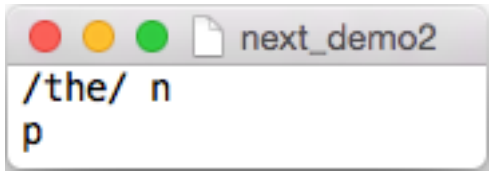
- Nesse exemplo é demonstrada a Próxima instrução, que quando processa a linha selecionada (linha 3), sed imediatamente inicia o processamento da próxima linha sem mostrá-la




```
$ sed -n -f next_demo1 lines
Line one.
The second line.
This is line four.
Five.
This is the sixth sentence.
This is line seven.
Eighth and last.
```

Próxima Instrução (n)

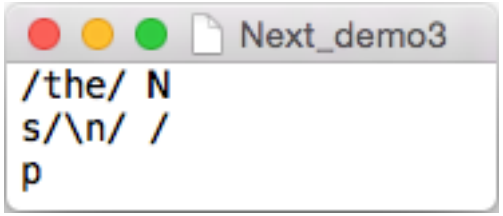
- Nesse exemplo é usado um seletor textual em todas as linhas que possuem o texto "the" (linha 6)




```
$ sed -n -f next_demo2 lines
Line one.
The second line.
The third.
This is line four.
Five.
This is line seven.
Eighth and last.
```

Próxima Instrução sem Escrita (N)

- Exemplo similar ao anterior, mas usando a instrução **N** ao invés de n que anexa a próxima linha na linha selecionada (com o texto "the") com uma nova linha (\n) entre eles; depois o programa substitui a nova linha por espaço, mostrando a linha 6 e 7 juntas




```
$ sed -n -f Next_demo3 lines
Line one.
The second line.
The third.
This is line four.
Five.
This is the sixth sentence. This is line seven.
Eighth and last.
```

MAIS EXEMPLOS

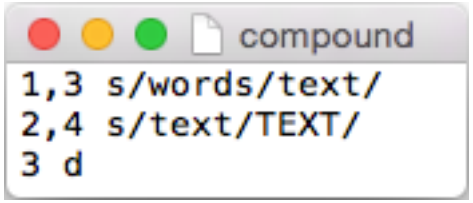
Outro Arquivo de Entrada

- Considere o seguinte arquivo de entrada para mostrar a execução de múltiplas instruções **sed**

```
$ cat > compound.in << EOF
> 1. The words on this page...
> 2. The words on this page...
> 3. The words on this page...
> 4. The words on this page...
> EOF
$
```

Múltiplas Instruções

- O programa a seguir substitui a string "words" por "text" nas linhas 1, 2 e 3 e a string "text" por "TEXT" nas linhas 2, 3 e 4, além de selecionar e remover a linha 3

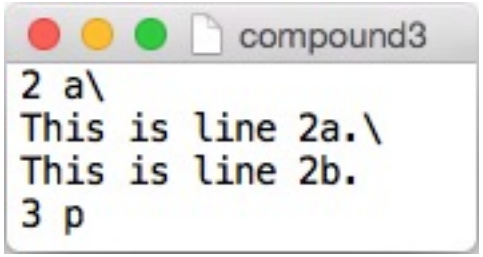



```
$ sed -f compound compound.in
1. The text on this page...
2. The TEXT on this page...
4. The words on this page...
$
```

O que acontece se inverter as linhas 1 e 2 do programa?

Múltiplas Instruções

- Outro exemplo onde o programa adiciona duas novas linhas à linha 2; a última instrução imprime a linha 3 repetida

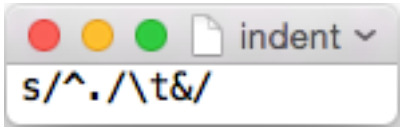



```
$ sed -f compound3 compound.in
1. The words on this page...
2. The words on this page...
This is line 2a.
This is line 2b.
3. The words on this page...
3. The words on this page...
4. The words on this page...
$
```

O que acontece se trocar a última instrução por "`2 d`"?

Expressões Regulares

- Esse exemplo usa expressões regulares no padrão; a expressão regular casa um caractere não vazio no começo da linha (^.) e substitui pela string entre as barras que adiciona uma tabulação e & que recebe o texto casado pela expressão regular



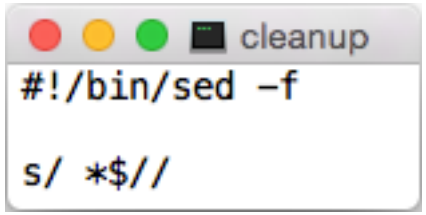
Cuidado: no Mac OS X foi preciso usar **$'\t'**

```
$ sed -f indent lines
    Line one.
    The second line.
    The third.
    This is line four.
    Five.
    This is the sixth sentence.
    This is line seven.
    Eighth and last.
$
```

Scripts *Standalone*

- Scripts podem executar o comando **`sed`** de dentro deles (executa a shell e depois o comando **`sed`**); pode-se evitar esse *overhead* criando um *script* sed
  - Para isso, basta colocar no cabeçalho **`#!/bin/sed -f`**

- Exemplo de um script autocontido que usa uma expressão regular para remover todos os caracteres em branco no final de linhas



Dica: como *scripts* só podem usar um parâmetro no *shebang*, se quiser usar a opção –n deve-se usar `-nf` (nessa ordem): #!/bin/sed –nf

Hold Space

- Esse exemplo faz uso do Hold space para trocar pares de linhas em um arquivo

s1

```
h # Copy Pattern space (line just read) to Hold space.
n # Read the next line of input into Pattern space.
p # Output Pattern space.
g # Copy Hold space to Pattern space.
p # Output Pattern space (which now holds the previous line).
```

```
$ sed -n -f s1 lines
The second line.
Line one.
This is line four.
The third.
This is the sixth sentence.
Five.
Eighth and last.
This is line seven.
$
```

Hold Space

- Adicionar uma linha em branco após cada linha do arquivo de entrada

```
$ sed 'G' lines
Line one.

The second line.

The third.

This is line four.

Five.

This is the sixth sentence.

This is line seven.

Eighth and last.

$
```

Hold Space

- Um programa para reverter um arquivo de entrada

s2

```
2,$G # On all but the first line, append a NEWLINE and the
     # contents of the Hold space to the Pattern space.
h    # Copy the Pattern space to the Hold space.
$!d  # Delete all except the last line.
```

```
$ sed -f s2 lines
Eighth and last.
This is line seven.
This is the sixth sentence.
Five.
This is line four.
The third.
The second line.
Line one.
$
```

Você entende o que esse programa está fazendo em cada passo?

ISSO É TUDO, PESSOAL!